| **Course:** | **Section:** |
| --- | --- |
| **Name/s:** | **Date Performed:** |
|  | **Date Submitted:** |
|  | **Instructor:** |
|  | |
| **Laboratory Activity No. 6** | |
| **Integrating Firebase Admin with NodeMCU** | |
|  | |
| **1. Objective(s):** | |
| This activity helps the student to explore the use of Firebase Admin with NodeMCU, focusing on real-time data management using Firebase Realtime Database or Firestore. Students will learn to securely transmit data from NodeMCU to a backend server that leverages Firebase Admin SDK, allowing robust interaction with Firebase services. | |
| **2. Intended Learning Outcomes (ILOs):** | |
| At the end of the activity student shall able to:   * Understand the Firebase Admin SDK’s role in managing data and authentication. * Configure NodeMCU to send data to a backend that securely communicates with Firebase. * Set up and deploy a simple Node.js server using Firebase Admin to handle data from IoT devices. * Securely store and retrieve data from Firebase using NodeMCU. | |
| **3. Materials** | |
| Hardware:   * NodeMCU ESP8266 or ESP32 module * DHT11 or other sensor (optional for data collection)   Software:   * Arduino IDE with ESP8266/ESP32 libraries installed * Firebase Admin SDK * Node.js and npm * Firebase Console * Postman (for testing)   Accounts and API Keys:   * Firebase project and database URL * Firebase Admin SDK service account key | |
| **4. Discussion** | |
| With the growth of IoT devices, managing real-time data securely is essential. The Firebase Admin SDK provides backend services to interact with Firebase Realtime Database, Firestore, Authentication, and other Firebase products without using Firebase directly on the client-side IoT devices, which can be less secure.  In this activity, students will configure NodeMCU to send sensor data to a Node.js backend server. This server, in turn, uses Firebase Admin SDK to write data securely to Firebase. This approach offers better control over data security, integrity, and scalability. | |
| **4. Procedures** | |
| **Step 1: Set Up Firebase Project and Database**   1. Create a Firebase Account:    1. Go to the Firebase Console.    2. Sign in with your Google account or create a new one. 2. Create a New Project:    1. Click on Add project.    2. Enter a name for your project (e.g., NodeMCU Project).    3. (Optional) Enable Google Analytics if needed, then click Create project. 3. Access Realtime Database or Firestore:    1. Once your project is created, go to the Build section in the left sidebar.    2. Choose **Firestore Database**.    3. For **Firestore** Database, click on Create Database and select the **Start in Test Mode** option. This allows read/write access without security rules for testing (make sure to secure it later). 4. Navigate to **Project Settings** > **Service accounts**. 5. Click **Generate new private key** to download the **JSON** file containing your service account key. This will be used in your Node.js backend.   **Step 2: Set Up Node.js Backend with Firebase Admin SDK**   1. Install Node.js and create a new Node.js project directory. 2. In the terminal, initialize your project:  | npm init -y | | --- |  1. Install necessary packages:  | npm install firebase-admin express body-parser | | --- |  1. Create a file named server.js and configure Firebase Admin SDK:  | const express = require('express');  const bodyParser = require('body-parser');  const admin = require('firebase-admin');  const serviceAccount = require('./path-to-your-service-account-file.json'); // Path to your service account key JSON file  // Initialize Firebase Admin SDK with Firestore  admin.initializeApp({  credential: admin.credential.cert(serviceAccount),  // No need for databaseURL when using Firestore  });  const db = admin.firestore(); // Use Firestore instead of Realtime Database  const app = express();  app.use(bodyParser.json());  app.post('/data', async (req, res) => {  const data = req.body;    try {  // Add data to Firestore in the 'sensorData' collection  await db.collection('sensorData').add(data);  res.status(200).send('Data saved successfully');  } catch (error) {  res.status(500).send(error);  }  });  const PORT = process.env.PORT || 3000;  app.listen(PORT, () => {  console.log(`Server is running on port ${PORT}`);  }); | | --- |  1. Save the code and run the server:  | node server.js | | --- |  1. Use Postman to test the endpoint by sending a POST request to http://localhost:3000/data with JSON data:  | {  "temperature": 22.5,  "humidity": 60  } | | --- |  1. Go to your Firebase Real-Time Database then you should see the changes in database value, like this:   **Step 3: Write the NodeMCU Code to Send Data to the Backend**   1. In Arduino IDE, add code to connect NodeMCU to WiFi and send data to your Node.js backend:  | #include <ESP8266WiFi.h>  #include <ESP8266HTTPClient.h>  const char\* ssid = "your-SSID";  const char\* password = "your-PASSWORD";  const char\* serverUrl = "http://<your-server-IP>:3000/data";  void setup() {  Serial.begin(115200);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(1000);  Serial.println("Connecting to WiFi...");  }  Serial.println("Connected to WiFi");  }  void loop() {  if (WiFi.status() == WL\_CONNECTED) {  WiFiClient client;  HTTPClient http;  http.begin(client, serverUrl);  http.addHeader("Content-Type", "application/json");  // Example data  String jsonData = "{\"temperature\": 25, \"humidity\": 60}";  int httpResponseCode = http.POST(jsonData);  if (httpResponseCode > 0) {  Serial.println("Data sent successfully!");  } else {  Serial.println("Error sending data.");  }  http.end();  }  delay(10000); // Send data every 10 seconds  } | | --- | |  |  1. Replace <your-SSID>, <your-PASSWORD>, and <your-server-IP> with actual values. 2. Upload the code to NodeMCU.   **Step 4: Check Data in Firebase**   1. Go to your Firebase Console and check the Realtime Database or Firestore. You should see data under the sensorData node. 2. Verify that new entries are being created as NodeMCU sends data every 10 seconds. | |
| **6. Supplemental Activity** | |
| **Supplemental Questions:**   1. Why do we use Firebase Admin on the server instead of accessing Firebase directly from the NodeMCU? 2. What are the advantages of using HTTPS for data transmission between NodeMCU and the server? 3. How can you secure the Firebase database rules to prevent unauthorized access?   **Supplemental Activity #1:**   1. Secure the Firebase credentials by storing them in the environmental variables.    1. install dot env using the command ***npm install doten***v in your terminal    2. Create a .env File in the Root of Your Vue.js Project. The .env file stores your sensitive information. Create the file with the following content:  | PORT=3000  FIREBASE\_DATABASE\_URL=https://<your-database-name>.firebaseio.com  FIREBASE\_SERVICE\_ACCOUNT=./path-to-your-service-account-file.json | | --- |  * 1. To prevent the .***env*** file from being committed to your version control system, add it to your .***gitignore*** file.   2. Modify the ***server.js*** file to use these environment variables:  | require('dotenv').config(); // Load environment variables  const express = require('express');  const bodyParser = require('body-parser');  const admin = require('firebase-admin');  // Load the service account key from the path stored in .env  const serviceAccount = require(process.env.FIREBASE\_SERVICE\_ACCOUNT);  admin.initializeApp({  credential: admin.credential.cert(serviceAccount),  databaseURL: process.env.FIREBASE\_DATABASE\_URL,  });  const db = admin.database();  const app = express();  app.use(bodyParser.json());  app.post('/data', (req, res) => {  const data = req.body;  db.ref('sensorData')  .push(data)  .then(() => res.status(200).send('Data saved successfully'))  .catch((error) => res.status(500).send(error));  });  const PORT = process.env.PORT || 3000;  app.listen(PORT, () => {  console.log(`Server is running on port ${PORT}`);  }); | | --- |  * 1. Now, when you run your application using node or nodemon, it shouldl load the environment variables from the .env file.  1. **Supplemental Activity #2: Sensor Integration and Data Visualization**    1. Connect a DHT11 or other environmental sensor to the NodeMCU.    2. Modify the NodeMCU code to send real sensor readings, such as temperature and humidity values, to the Node.js backend. 2. **Supplemental Activity #3: Develop a Vue.js Component to Display Firebase Data in Real-Time**    1. In your VueJs App, install Firebase SDK  | npm install firebase | | --- |  * 1. Configure Firebase in Vue by creating a Firebase config file ***firebaseConfig.js*** inside the src folder:  | // src/firebaseConfig.js  // src/firebaseConfig.js  import firebase from 'firebase/app';  import 'firebase/firestore';  const firebaseConfig = {  apiKey: "YOUR\_API\_KEY",  authDomain: "YOUR\_PROJECT\_ID.firebaseapp.com",  projectId: "YOUR\_PROJECT\_ID",  storageBucket: "YOUR\_PROJECT\_ID.appspot.com",  messagingSenderId: "YOUR\_MESSAGING\_SENDER\_ID",  appId: "YOUR\_APP\_ID"  };  // Initialize Firebase  firebase.initializeApp(firebaseConfig);  // Export the Firestore instance  const db = firebase.firestore();  export { db }; | | --- |  * 1. Create the Vue Component to Display Firebase Data. This can be done by creating new Vue component named ***SensorData.vue***  | <template>  <div>  <h1>Sensor Data</h1>  <ul>  <li v-for="data in sensorData" :key="data.id">  Temperature: {{ data.temperature }}°C, Humidity: {{ data.humidity }}%  </li>  </ul>  </div>  </template>  <script>  import { db } from '../firebase'; // Adjust the path as needed  export default {  data() {  return {  sensorData: [],  };  },  created() {  this.fetchData();  },  methods: {  async fetchData() {  try {  const snapshot = await db.collection('sensorData').get();  this.sensorData = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));  } catch (error) {  console.error("Error fetching sensor data: ", error);  }  },  },  };  </script>  <style scoped>  /\* Add any styles you need \*/  </style> | | --- |   The Vue component above is designed to fetch and display sensor data, specifically temperature and humidity, from a Firestore collection named sensorData. When the component is created, it automatically calls the fetchData method, which retrieves all documents from the specified Firestore collection using an asynchronous call. The retrieved data is then processed to create an array of objects that include each document's ID alongside its temperature and humidity values. This array is stored in the component's reactive state (sensorData), allowing the component to dynamically update the displayed information. The template renders this data in an unordered list format, ensuring that each data point is clearly presented to the user. Additionally, the component includes error handling to log any issues that occur during the data-fetching process, enhancing reliability and ease of debugging. Overall, the component effectively integrates real-time data retrieval with user-friendly presentation.   * 1. Update App.vue to Use the Component. In your App.vue, update the code by adding the Sensor Data component.  | <template>  <SensorData />  </template>  <script>  import SensorData from "./components/SensorData.vue";  export default {  name: "App",  components: {  SensorData,  },  };  </script> | | --- |  * 1. Run the application by starting the Vue Server.  | npm run dev | | --- |  1. **Optional - Notification Feature:** Implement Firebase Cloud Messaging (FCM) to send notifications to a mobile app or another client when certain sensor thresholds are reached. | |
| **7. Conclusion:** | |
| **What conclusion can you make based on the activity?**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | |
| **8. Assessment (Rubric for Activity):** | |
| | **Intended Learning Outcomes** | **Unsatisfactory**  **1** | **Satisfactory**  **2** | **Exemplary**  **3** | **Score** | | --- | --- | --- | --- | --- | | Configure theNodeMCU for Secure Data Transmission | NodeMCU is fully configured for secure data transmission with clear, secure communication to the backend. | NodeMCU is configured for data transmission with minor security gaps or setup issues. | NodeMCU is fully configured for secure data transmission with clear, secure communication to the backend. |  | | Setup and Deploy the Node.js Backend with Firebase Admin | Backend is fully deployed and securely communicates with Firebase, handling all data from IoT devices. | Backend is deployed and communicates with Firebase, but there are minor issues in handling data. | Backend is fully deployed and securely communicates with Firebase, handling all data from IoT devices. |  | | Data Storage and Retrieval from Firebase | Data is securely stored and retrieved from Firebase with effective handling of incoming NodeMCU data. | Data is stored and retrieved with minor gaps in security or data handling. | Data is securely stored and retrieved from Firebase with effective handling of incoming NodeMCU data. |  | | **Other comments/observation:** | | Total Score | |  | | RATING = (total score) x 100%  9 | |  |   Evaluated by: Date:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Printed Name and Signature of Faculty Member | |